Exploratory Data Analysis on Customer Churn

WHAT IS CHURN ANALYSIS ?

Customer churn analysis refers to identifying customers who are likely to stop using a company’s product or service. It helps in :

- Understanding why customers leave.
- Predicting future churn.
- Creating retention strategies.

TOOLS & TECHNOLOGIES USED

- **Jupyter Notebook:** To execute code and display results.
- **Python:** A programming language for data analysis.
- **Libraries Used:** Numpy, Pandas, Matplotlib, and Seaborn.
- **CSV Dataset:** "Customer Churn.csv"

BUSINESS REQUIREMENTS

1. What percentage of customers have churned?
2. Is churn related to gender or age group (senior citizen)?
3. How does customer tenure affect churn?
4. Which contract type leads to higher churn?
5. Does the use of specific services (e.g., online security, tech support) impact churn?
6. Do payment methods influence churn rate?
7. Are there data quality issues that need cleaning before analysis?

Importing Required Libraries

[1]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from anyio.abc import value
```

“We import essential Python libraries for data loading, manipulation, and visualization.”

Loading the Dataset and Displaying Initial Rows

[11]:
```
df=pd.read_csv(r"C:\Users\sunit\Documents\DATA ANALYSTS\Python\Churn Analysis Project\Customer Churn.csv")
df.head(5)
```

[11]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	St
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	

5 rows × 21 columns

“We load the churn dataset into a pandas DataFrame to begin our analysis and preview the first few records to understand the structure and data types.”

Inspecting Data Information

```
[12]: #data info
      df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
     viceProtection    7043 non-null   object
     TechSupport       7043 non-null   object
```

Expand Output

"Checking column data types, non-null values, and overall structure of the dataset."

Handling Null and Inconsistent Data

```
[13]: # lets convert totalcharges null rows into 0 and change its data type from object to float
df["TotalCharges"]= df["TotalCharges"].replace(" ","0")
df["TotalCharges"]= df["TotalCharges"].astype("float")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

“Fixing missing or incorrectly formatted values in the ‘TotalCharges’ column.”

Handling Null and Inconsistent Data

```
[13]: # lets convert totalcharges null rows into 0 and change its data type from object to float
df["TotalCharges"]= df["TotalCharges"].replace(" ","0")
df["TotalCharges"]= df["TotalCharges"].astype("float")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

“Fixing missing or incorrectly formatted values in the ‘TotalCharges’ column.”

Checking for Null Values

[17]:
```
#let's check null values in our data
df.isnull().sum()
```

[17]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

[14]:
```
#and if we add one more .sum() it will sum it all
df.isnull().sum().sum()
```

[14]:
```
0
```

“Fixing missing or incorrectly formatted values in the ‘TotalCharges’ column.”

Checking for Duplicate Records

```
[20]: df.duplicated().sum()

[20]: 0

[21]: #lets check for customerID as it has unique values
      df["customerID"].duplicated().sum()

[21]: 0
```

“Verifying that customer records are unique and there are no duplicates.”

Descriptive Statistics

```
[15]: df.describe()
```

[15]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

“Getting summary statistics (mean, std, min, max) for numeric columns.”

Converting Numeric to Categorical

```
def conv(value):
    if value == 1:
        return "Yes"
    else:
        return "No"

df["SeniorCitizen"] = df["SeniorCitizen"].apply(conv)
df.head(50)
```

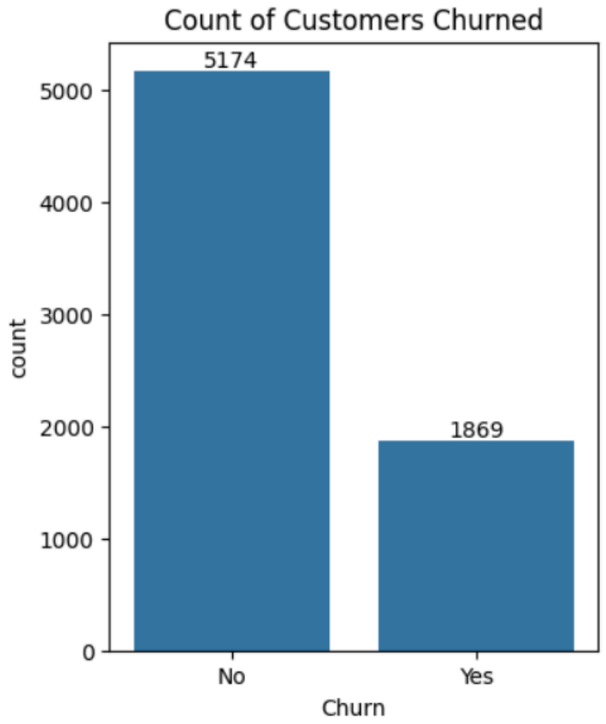
[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSuppo
0	7590-VHVEG	Female	No	Yes	No	1	No	No phone service	DSL	No	...	No	N
1	5575-GNVDE	Male	No	No	No	34	Yes	No	DSL	Yes	...	Yes	N
2	3668-QPYBK	Male	No	No	No	2	Yes	No	DSL	Yes	...	No	N
3	7795-CFOCW	Male	No	No	No	45	No	No phone service	DSL	Yes	...	Yes	Y
4	9237-HQITU	Female	No	No	No	2	Yes	No	Fiber optic	No	...	No	N
5	9305-CDSKC	Female	No	No	No	8	Yes	Yes	Fiber optic	No	...	Yes	N

“Making ‘SeniorCitizen’ easier to interpret by converting numeric to 'Yes/No'.”

Count of Churned Customers

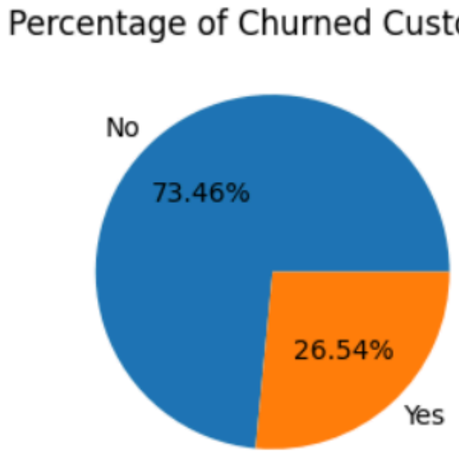
```
[47]: #let's check out how many customer churned out
plt.figure(figsize=(4,5))
ax = sns.countplot(x="Churn", data = df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customers Churned")
plt.show()
```



“Visualizing how many customers have churned vs. retained.”

Churn Percentage

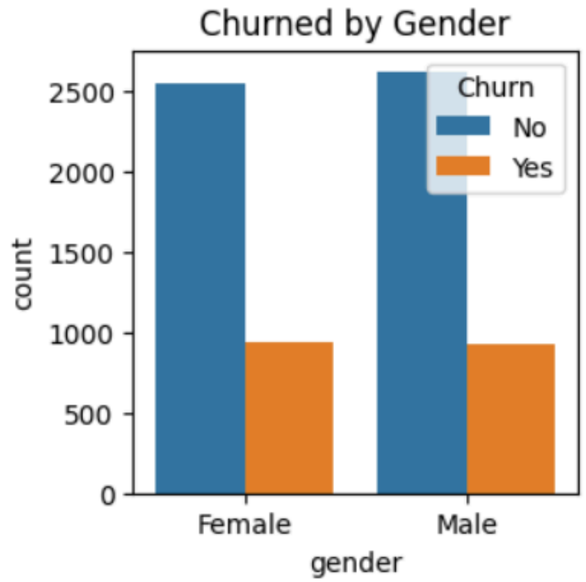
[43]:
```
#Let's check the percentage of customers churned
plt.figure(figsize=(3,3))
gb = df.groupby("Churn").agg({"Churn":"count"})
plt.pie(gb["Churn"], labels = gb.index, autopct="%1.2f%%")
plt.title("Percentage of Churned Customer")
plt.show()
```



“Visualizing the percentage of churned customers using a pie chart.”

Churn by Gender

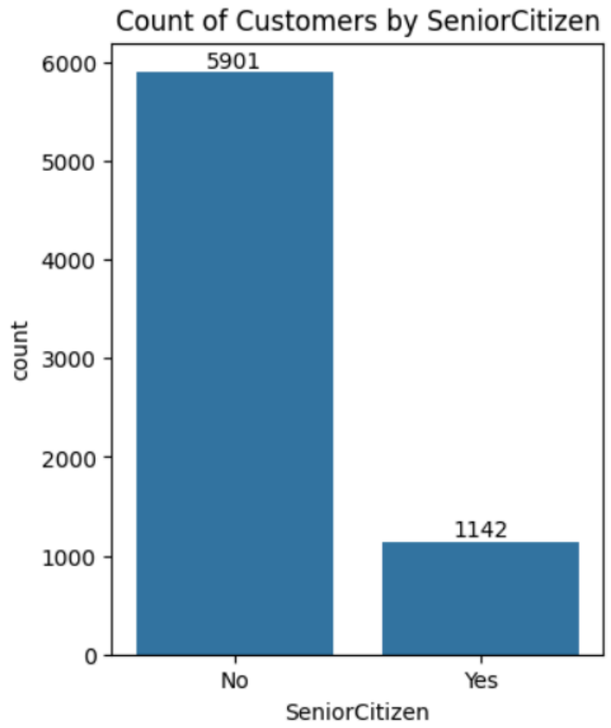
[42]:
```
#let's explorer the reason behind the customer churned through gender
plt.figure(figsize=(3,3))
sns.countplot(x="gender",data = df,hue= "Churn")
plt.title("Churned by Gender")
plt.show()
```



“Analyzing if churn behavior varies by gender.”

Count of Senior Citizen

[6]:
```
#Lets count senior citizen
plt.figure(figsize=(4,5))
ax = sns.countplot(x="SeniorCitizen", data = df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by SeniorCitizen")
plt.show()
```



“Exploring how many senior citizen customers we have.”

Churn Analysis by Senior Citizen

```
[9]: plt.figure(figsize=(3,3))
     sns.countplot(x="SeniorCitizen",data = df,hue= "Churn")
     plt.title("Churned by SeniorCitizen")
     plt.show()
```

“Grouped bar chart showing churn comparison between senior and non-senior customers.”

Churn vs. Tenure

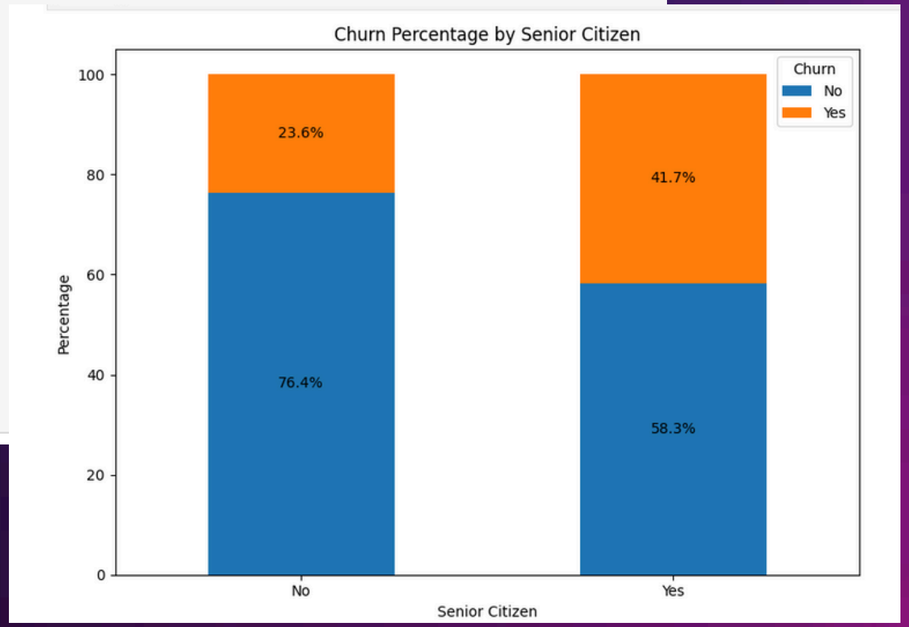
•[10]:

```
#Stacked bar chart with labels as %
grouped = df.groupby(["SeniorCitizen", "Churn"]).size().unstack().fill na(0)
percent = grouped.div(grouped.sum(axis=1), axis=0) * 100  # convert to percentages

#Step 1: Calculate percentage data
grouped = df.groupby(['SeniorCitizen', 'Churn']).size().unstack(fill_value=0)

#Step 2: Convert to percentages
percentages = grouped.div(grouped.sum(axis=1), axis=0) * 100

#Step 3: Plot
ax = percentages.plot(kind='bar', stacked=True, figsize=(8,6))

#Step 4: Add labels on bars
for i, row in enumerate(percentages.values):
    cumulative = 0
    for j, val in enumerate(row):
        if val > 0:
            plt.text(i, cumulative + val/2, f'{val:.1f}%', ha='center', va='center', fontsize=10)
            cumulative += val
```

“Stacked bar showing churn percentage among senior and non-senior citizens for comparison.”

Churn vs. Tenure

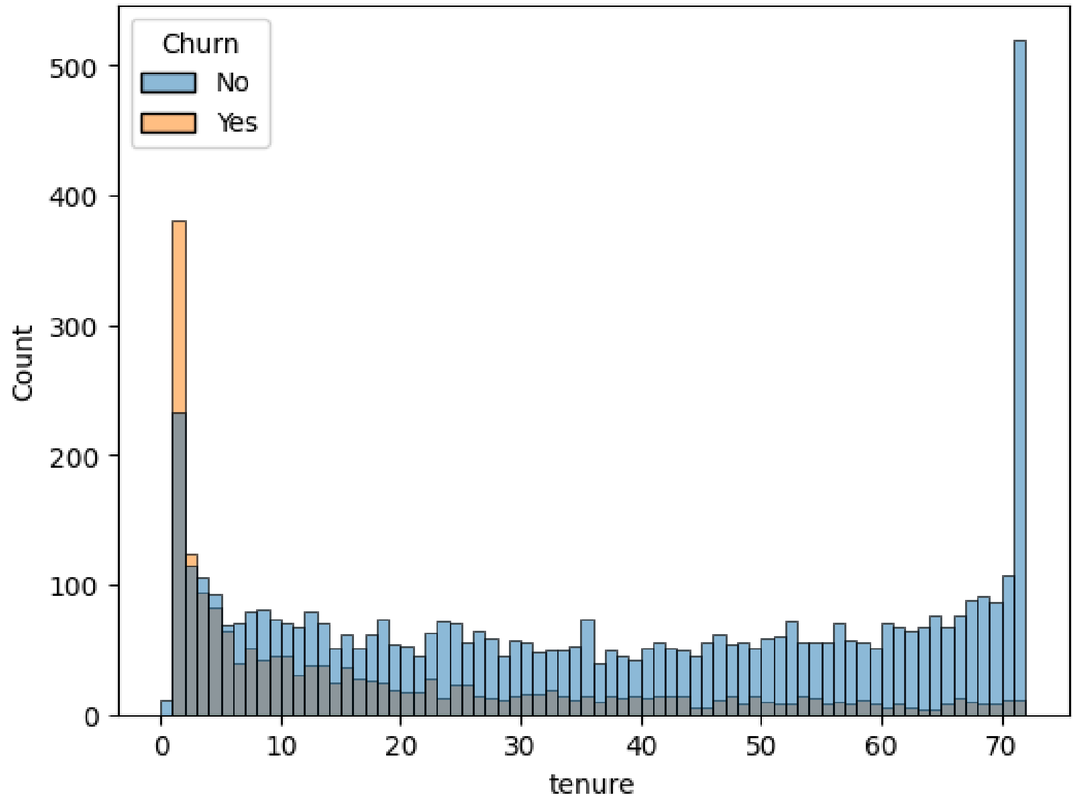
```python
#Step 4: Add labels on bars
for i, row in enumerate(percentages.values):
    cumulative = 0
    for j, val in enumerate(row):
        if val > 0:
            plt.text(i, cumulative + val/2, f'{val:.1f}%', ha='center', va='center', fontsize=10)
            cumulative += val

plt.title("Churn Percentage by Senior Citizen")
plt.ylabel("Percentage")
plt.xlabel("Senior Citizen")
plt.xticks(rotation=0)
plt.legend(title="Churn")
plt.tight_layout()
plt.show()
```



“Stacked bar showing churn percentage among senior and non-senior citizens for comparison.”

Churn by Contract Type

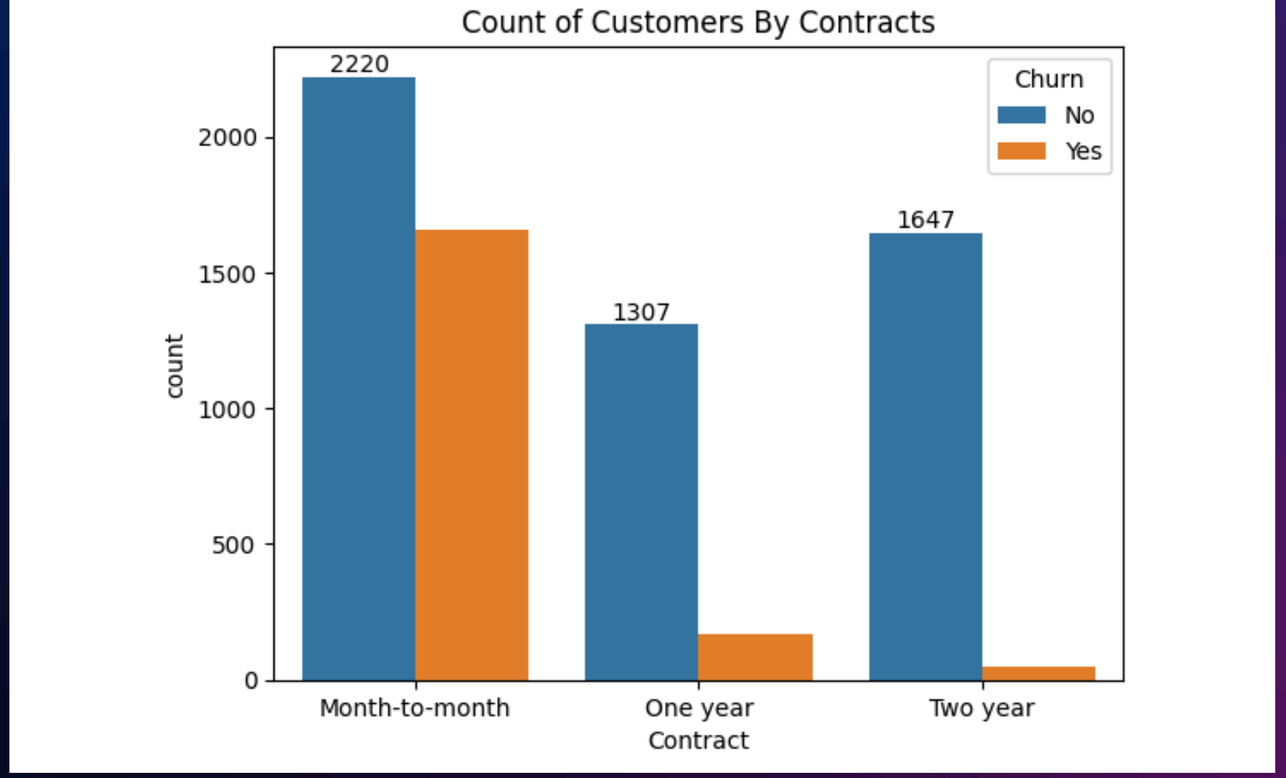
```
[31]: # Let's check the churned rate on the basis of tenure
sns.histplot(x="tenure",data=df,hue="Churn",bins=72)
plt.show()
```



“Understanding how contract duration impacts churn.”

Churn by Services Used

```
[32]: #Let's count the customers on the basis of contract.
ax = sns.countplot(x="Contract", data = df,hue="Churn")
ax.bar_label(ax.containers[0])
plt.title("Count of Customers By Contracts")
plt.show()
```



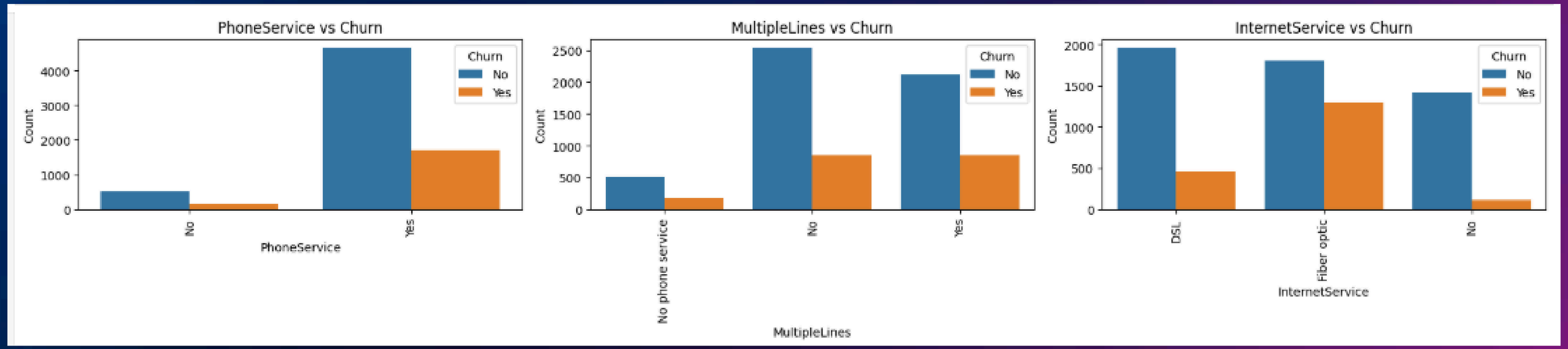
“Churn analysis by contract type to see which contract leads to higher customer loss.

Churn by Available Services

[35]:
```python
# Lets create subplots for all the services provide to the customers and check its churn rate
# List of columns you want to plot
cols = [
    'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
    'OnlineBackup', 'DeviceProtection', 'TechSupport',
    'StreamingTV', 'StreamingMovies'
]

# Set up the subplot grid
n_cols = 3  # Number of columns in subplot grid
n_rows = (len(cols) + n_cols - 1) // n_cols  # Auto-adjust rows

plt.figure(figsize=(18, n_rows * 4))  # Adjust figure size based on rows

# Loop through columns and create countplots
for idx, col in enumerate(cols, 1):
    plt.subplot(n_rows, n_cols, idx)
    sns.countplot(x=col, data=df, hue='Churn')
    plt.title(f'{col} vs Churn')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.xticks(rotation=90)

plt.tight_layout()
plt.show()
```

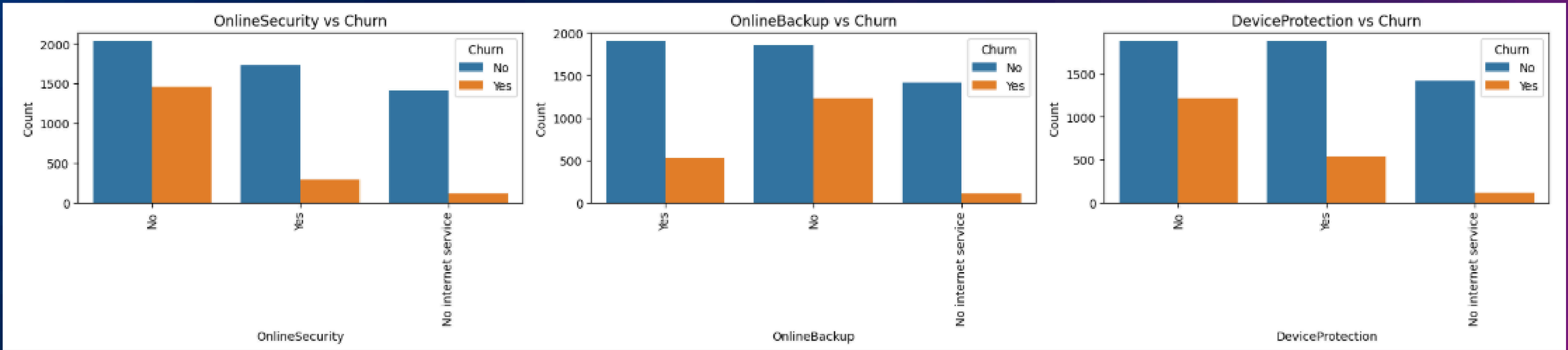
“Churn comparison across various customer services to identify which services are linked to higher churn rates.”

Churn by Available Services



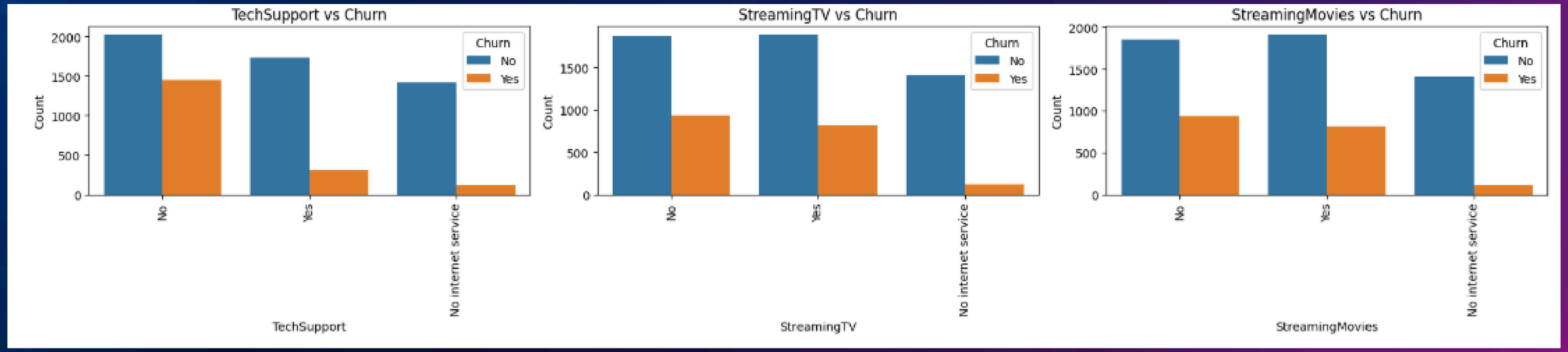
“Churn comparison across various customer services to identify which services are linked to higher churn rates.”

Churn by Available Services



“Churn comparison across various customer services to identify which services are linked to higher churn rates.”

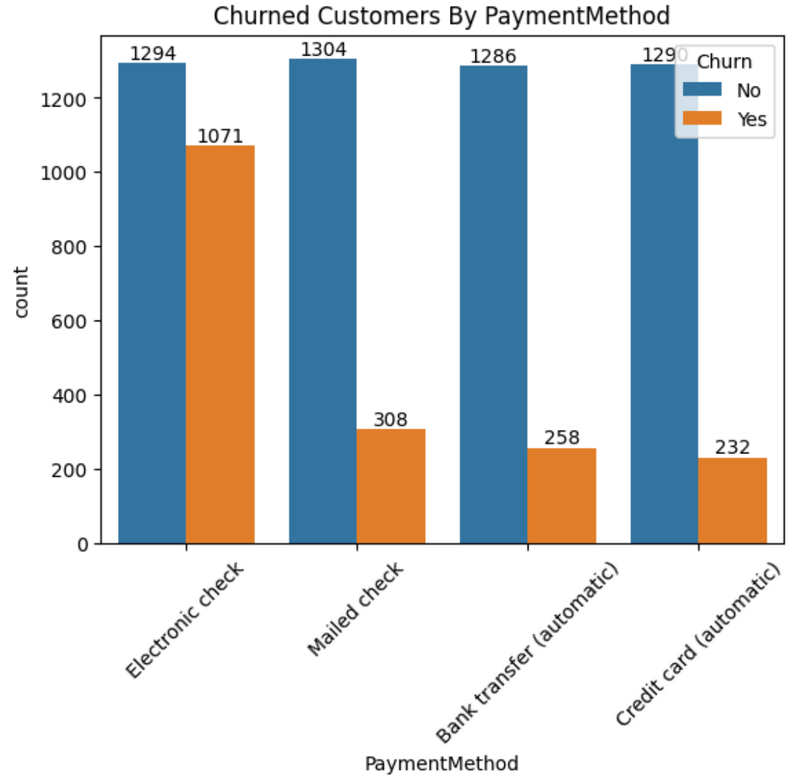
Churn by Available Services



“Churn comparison across various customer services to identify which services are linked to higher churn rates.”

Churn by Payment Method

```
#Let's count the churned customers on the basis of payment method.
ax = sns.countplot(x="PaymentMethod", data = df,hue="Churn")
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.title("Churned Customers By PaymentMethod")
plt.xticks(rotation=45)
plt.show()
```



“Grouped bar chart showing churn distribution across different payment methods.”

KEY FINDINGS FOR CUSTOMER RETENTION

1. **High Churn:** A **26.54% churn rate (1,869 customers)** demands urgent attention.
2. **Gender-Neutral Churn:** Churn rates are **consistent across genders**, suggesting the need for universal retention efforts.
3. **Senior Citizen Loyalty: Senior citizens churn less**, indicating effective engagement strategies in this demographic.
4. **Contract Vulnerability: Month-to-month contracts carry the highest risk of churn**; **long-term contracts** significantly reduce churn.
5. **Service Gaps & Churn:** Customers are more likely to churn if they **lack extra services** (like online security or tech help) or if they have **fiber-optic internet**.
6. **Payment Method Risk: Electronic Check users** show a higher churn rate, requiring investigation.

RECOMMENDATIONS

1. **Prioritize Contract Type:** Focus retention efforts on customers on month-to-month contracts by offering incentives for longer-term commitments.
2. **Enhance Service Bundling:** Promote the adoption of value-added services (OnlineSecurity, TechSupport, etc.) to increase customer stickiness and perceived value.
3. **Investigate Fiber Optic Churn:** Conduct deeper analysis into Fiber Optic customer feedback and service quality to identify and address pain points.
4. **Optimize Electronic Check Experience:** Analyze the reasons behind higher churn among Electronic Check users; consider alternative payment method promotions or improving the electronic check process.
5. **Leverage Senior Citizen Success:** Understand what makes senior citizens less likely to churn and explore if these strategies can be applied to other demographics.

THANK YOU